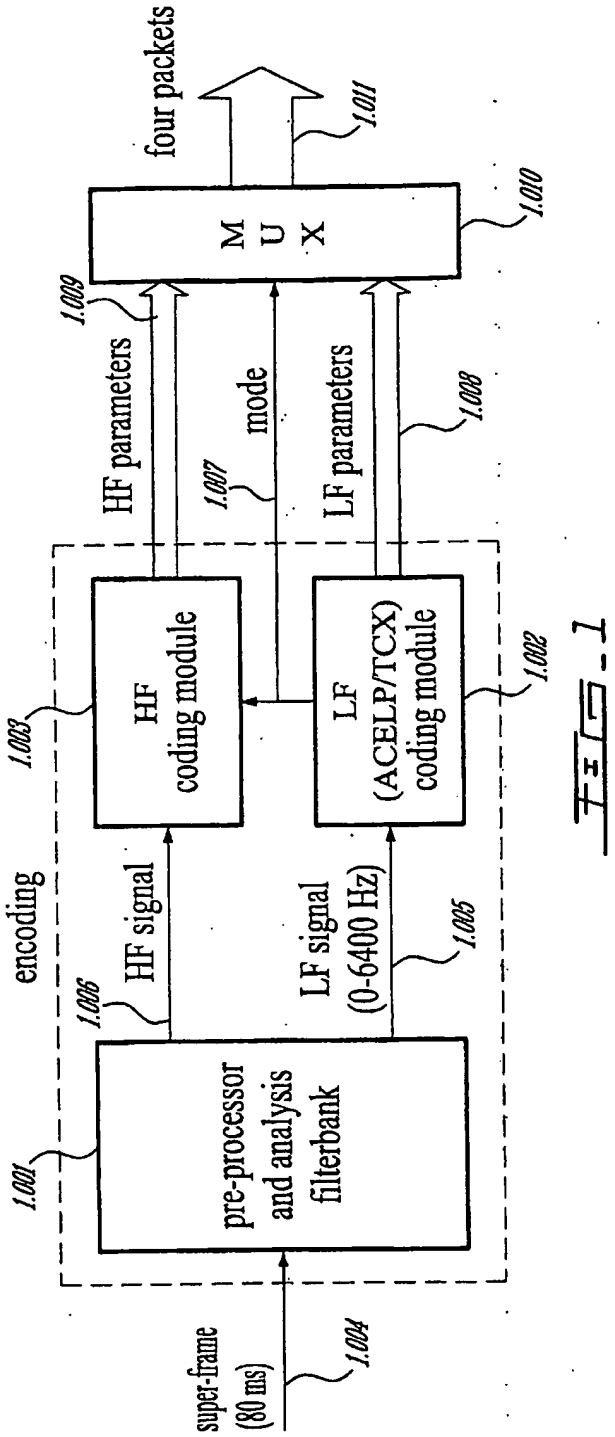

four packets
1.011
M U X
1.010
1.009
HF parameters
mode
1.007
LF parameters
1.008
HF coding module
1.003
LF (ACELP/TCX) coding module
1.002
encoding
HF signal
1.006
LF signal (0-6400 Hz)
1.005
pre-processor and analysis filterbank
1.001
super-frame (80 ms)
1.004

FIG. 1

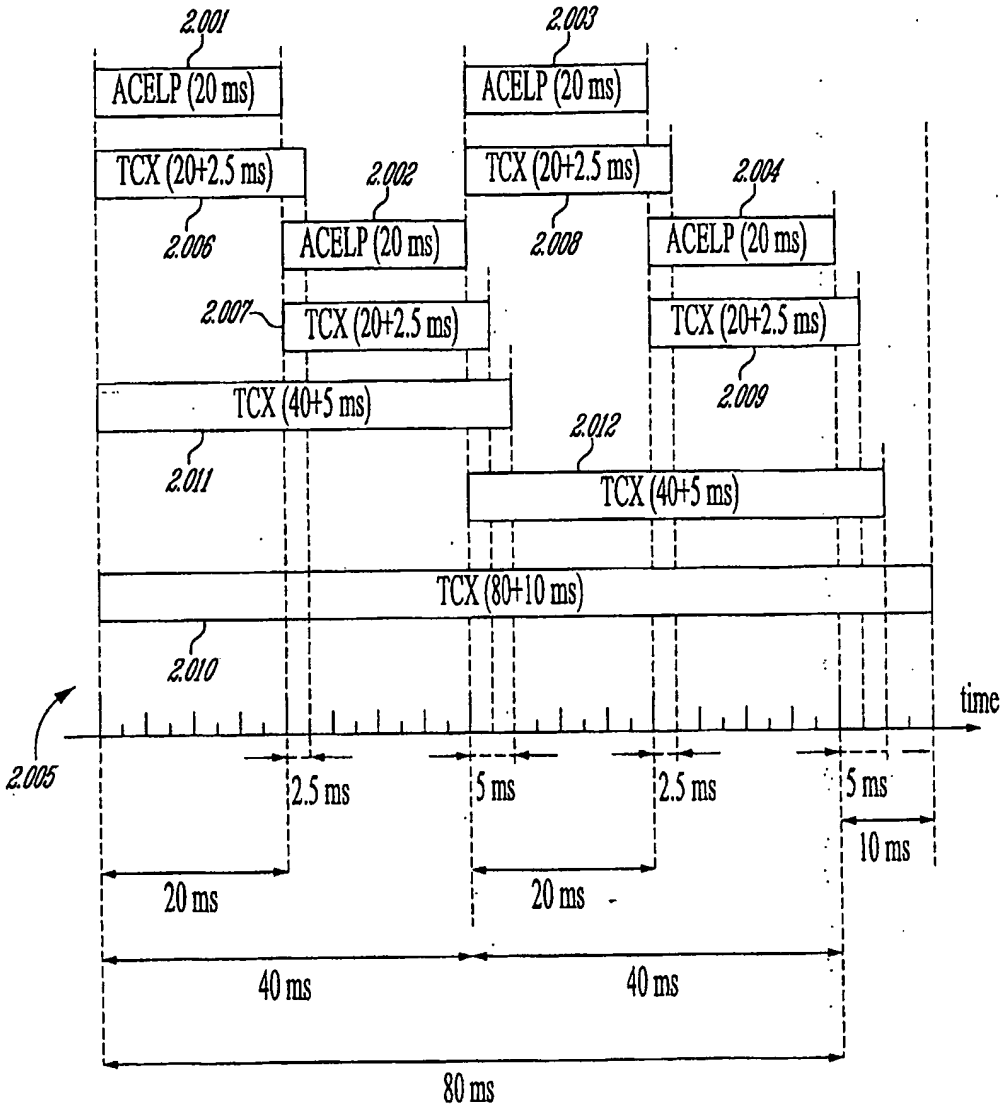

2.001
ACELP (20 ms)
2.003
ACELP (20 ms)
TCX (20+2.5 ms)
2.002
TCX (20+2.5 ms)
2.004
2.006
ACELP (20 ms)
2.008
ACELP (20 ms)
2.007
TCX (20+2.5 ms)
TCX (20+2.5 ms)
2.009
TCX (40+5 ms)
2.012
2.011
TCX (40+5 ms)
TCX (80+10 ms)
2.010
time
2.005
2.5 ms
5 ms
2.5 ms
5 ms
10 ms
20 ms
20 ms
40 ms
40 ms
80 ms

FIG. 2

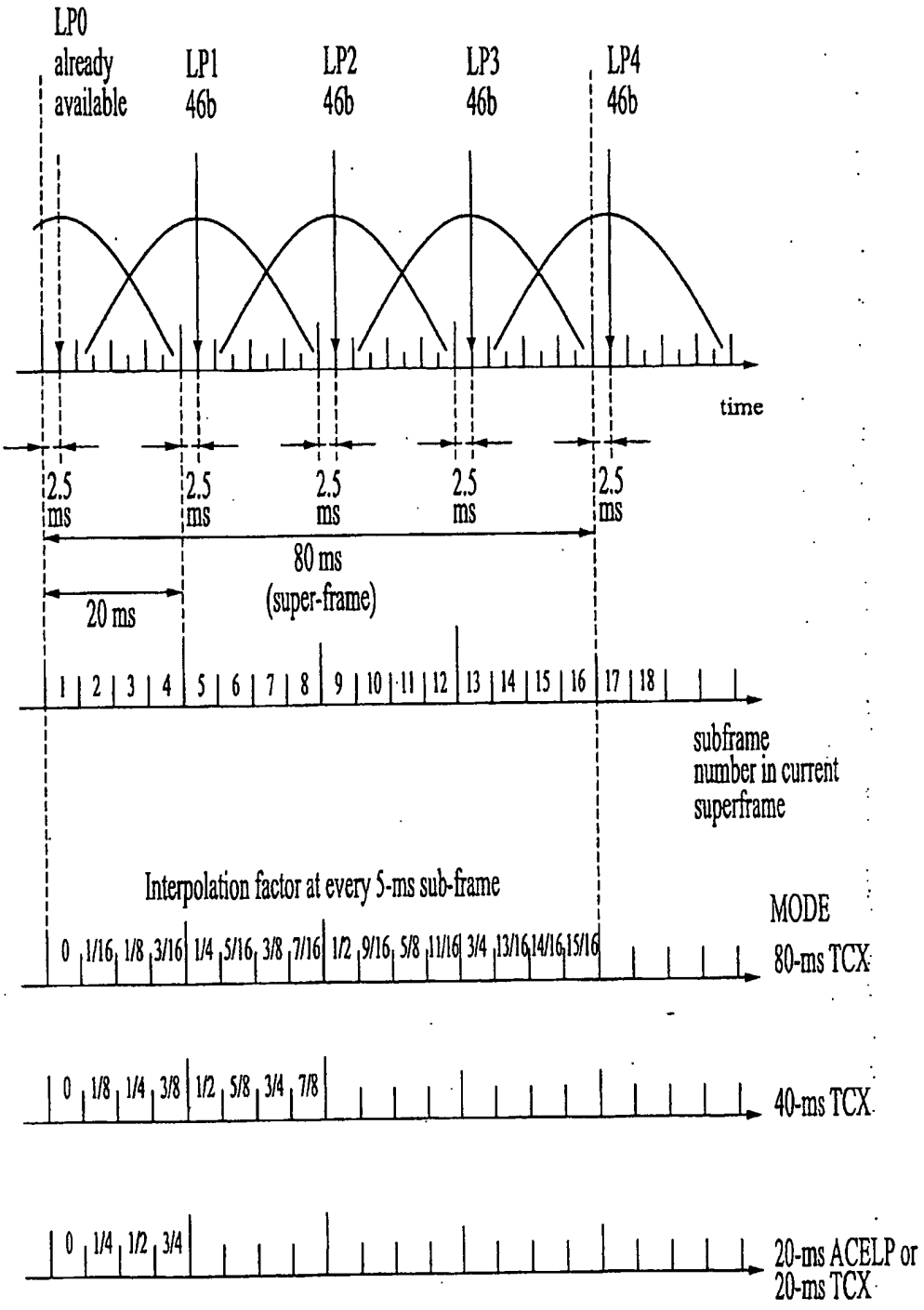
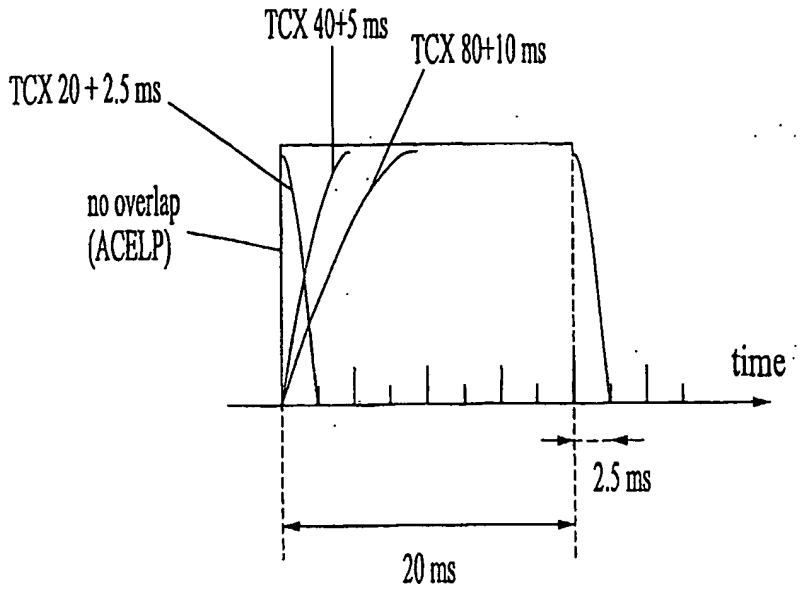
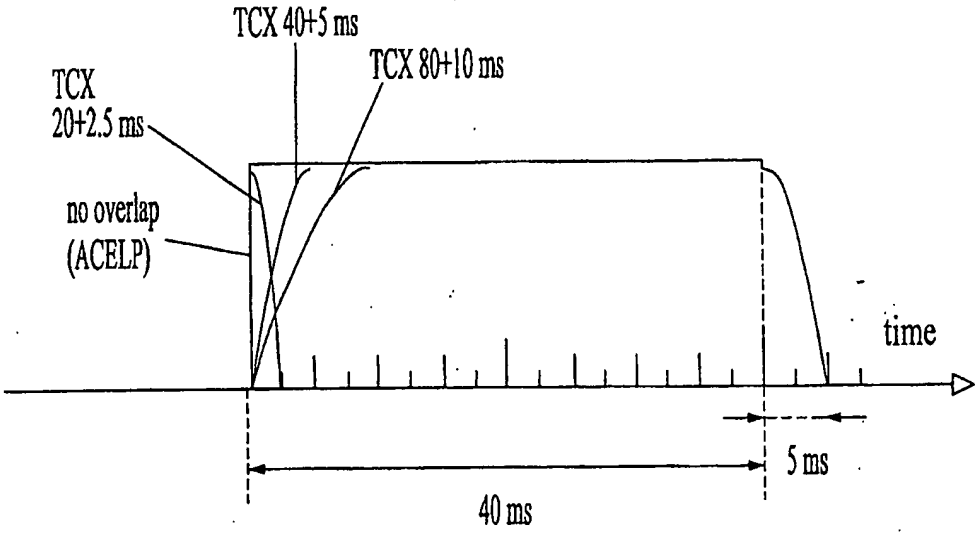

LP0
already
available
LP1
46b
LP2
46b
LP3
46b
LP4
46b
time
2.5
ms
2.5
ms
2.5
ms
2.5
ms
2.5
ms
80 ms
(super-frame)
20 ms
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
subframe
number in current
superframe
Interpolation factor at every 5-ms sub-frame
MODE
0 1/16 1/8 3/16 1/4 5/16 3/8 7/16 1/2 9/16 5/8 11/16 3/4 13/16 14/16 15/16
80-ms TCX
0 1/8 1/4 3/8 1/2 5/8 3/4 7/8
40-ms TCX
0 1/4 1/2 3/4
20-ms ACELP or
20-ms TCX

FIG. 3



windowing in TCX 20+2.5 ms

FIG_4A



windowing in TCX 40+5 ms

FIG_4B

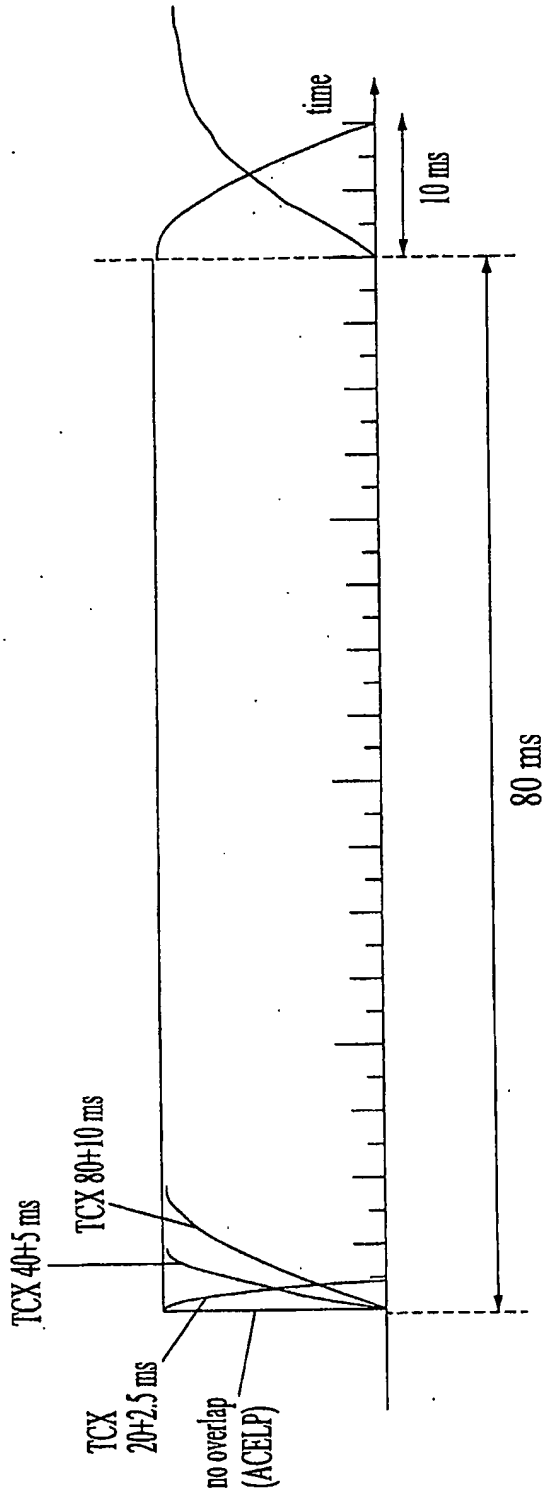

time
10 ms
80 ms
in TCX 80+10 ms
TCX 80+10 ms
TCX 40+5 ms
TCX
20+2.5 ms
no overlap
(ACELP)

FIG. 4C

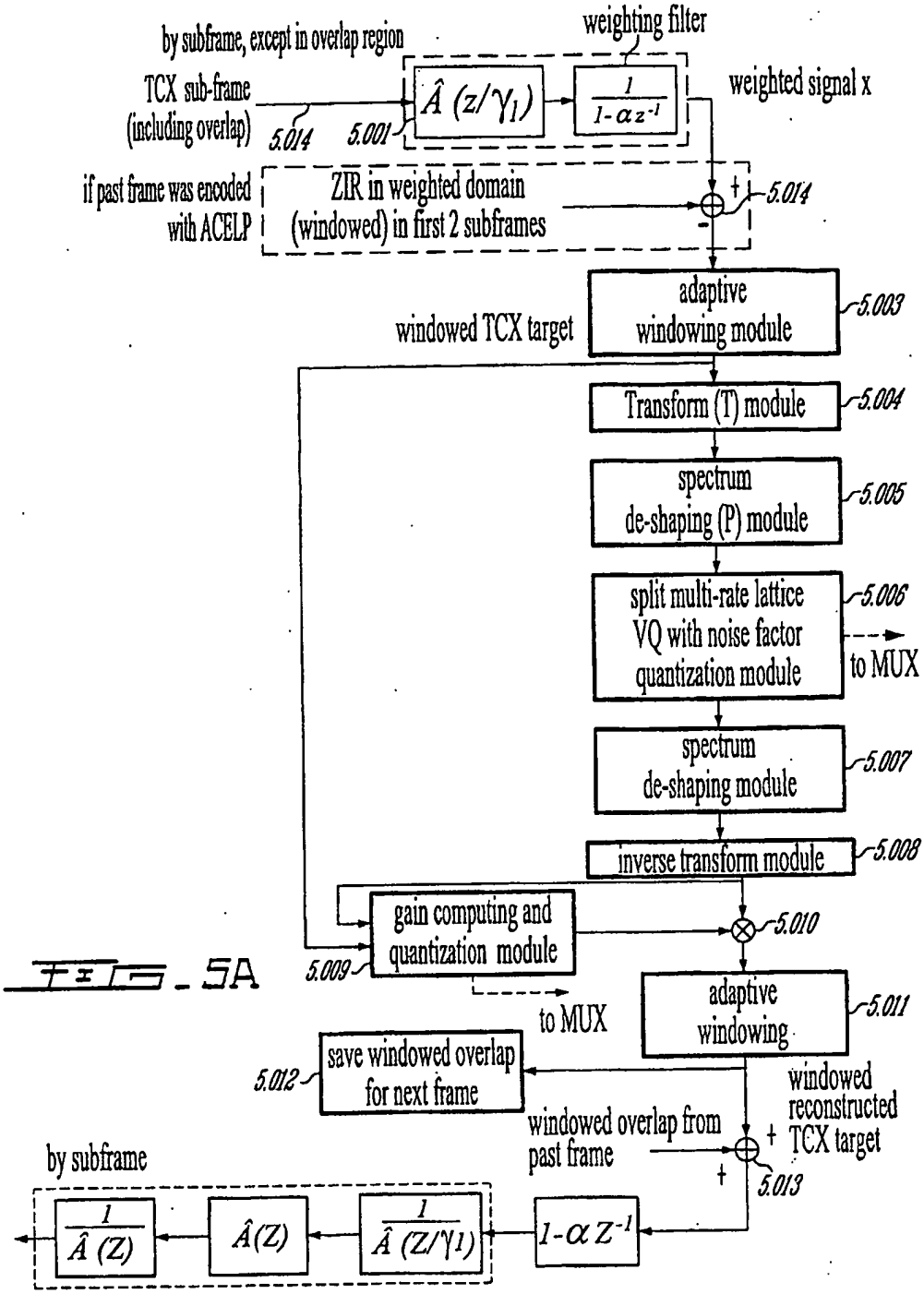


FIG. 5A

states initialized to 0 in $\frac{1}{\hat{A}(Z/\gamma_1)}$ and $1-\alpha Z^{-1}$ if past frame was encoded with ACELP

state initialized to past synthesis in $\frac{1}{\hat{A}(Z/\gamma_1)}$ otherwise

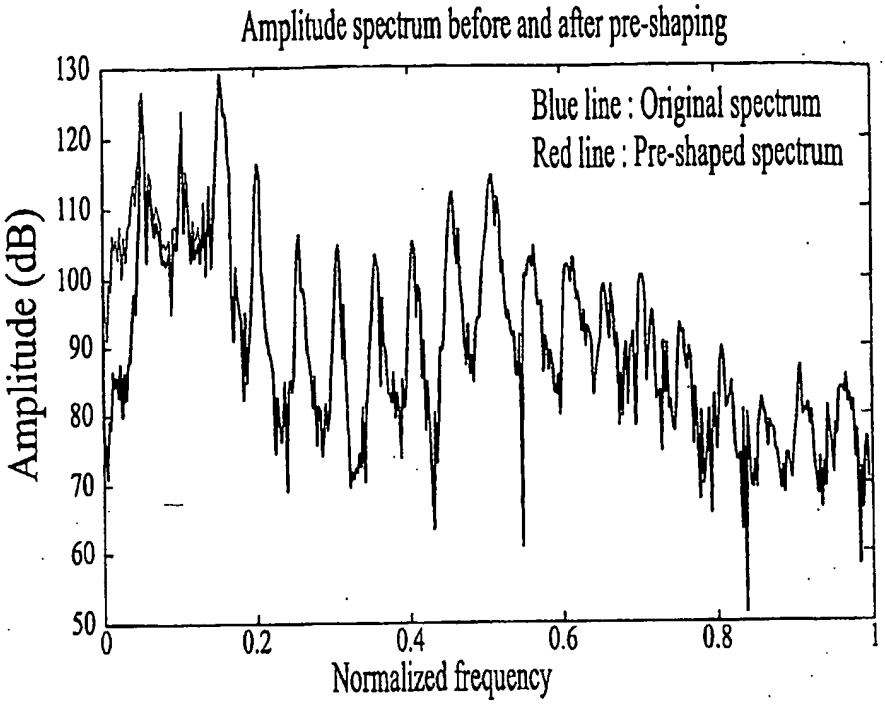

Amplitude spectrum before and after pre-shaping
Blue line : Original spectrum
Red line : Pre-shaped spectrum
Amplitude (dB)
130
120
110
100
90
80
70
60
50
0
0.2
0.4
0.6
0.8
1
Normalized frequency

FIG. 5B

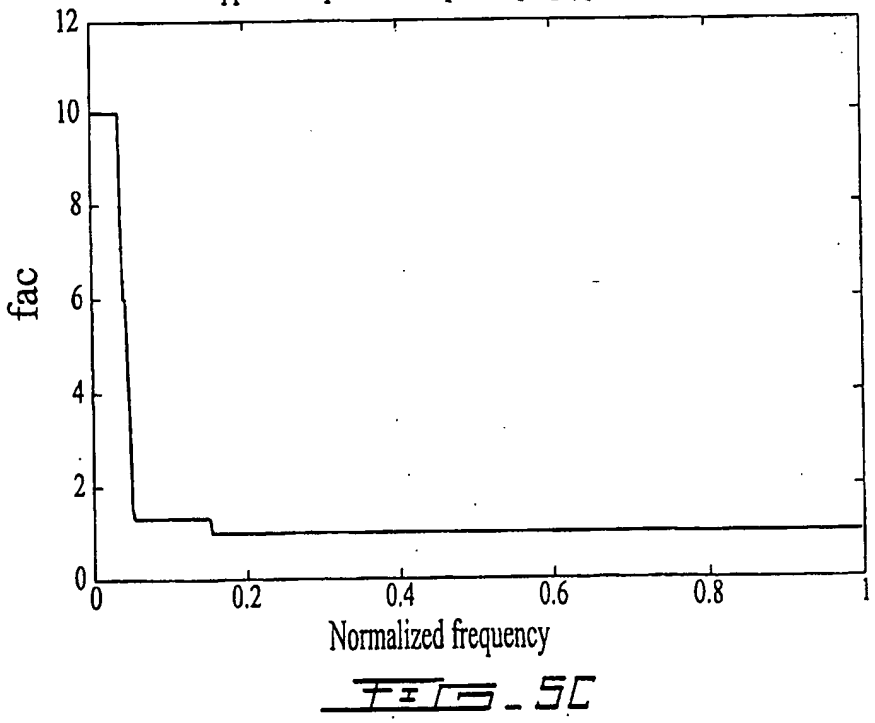

Gain applied to spectrum for pre-shaping (specific for this example)
fac
12
10
8
6
4
2
0
0
0.2
0.4
0.6
0.8
1
Normalized frequency

FIG. 5C

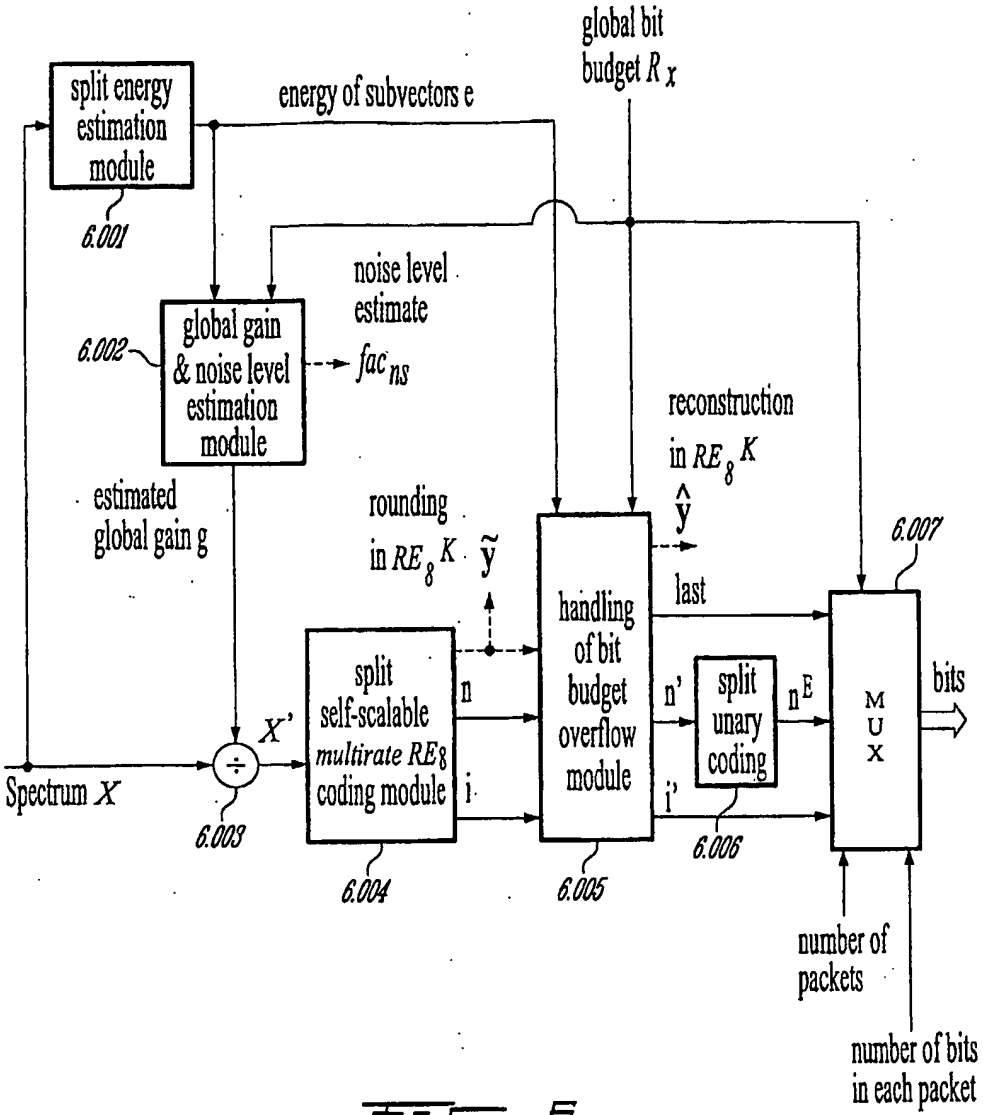

global bit budget R_x
split energy estimation module
energy of subvectors e
6.001
noise level estimate
global gain & noise level estimation module
6.002
fac_{ns}
reconstruction in $RE_8 K$
$\hat{y}$
estimated global gain g
rounding in $RE_8 K$
$\tilde{y}$
6.007
handling of bit budget overflow module
last
split self-scalable multirate RE_8 coding module
n
n'
split unary coding
n^E
bits
M U X
X'
Spectrum X
i
i'
6.003
6.004
6.005
6.006
number of packets
number of bits in each packet

FIG. 6

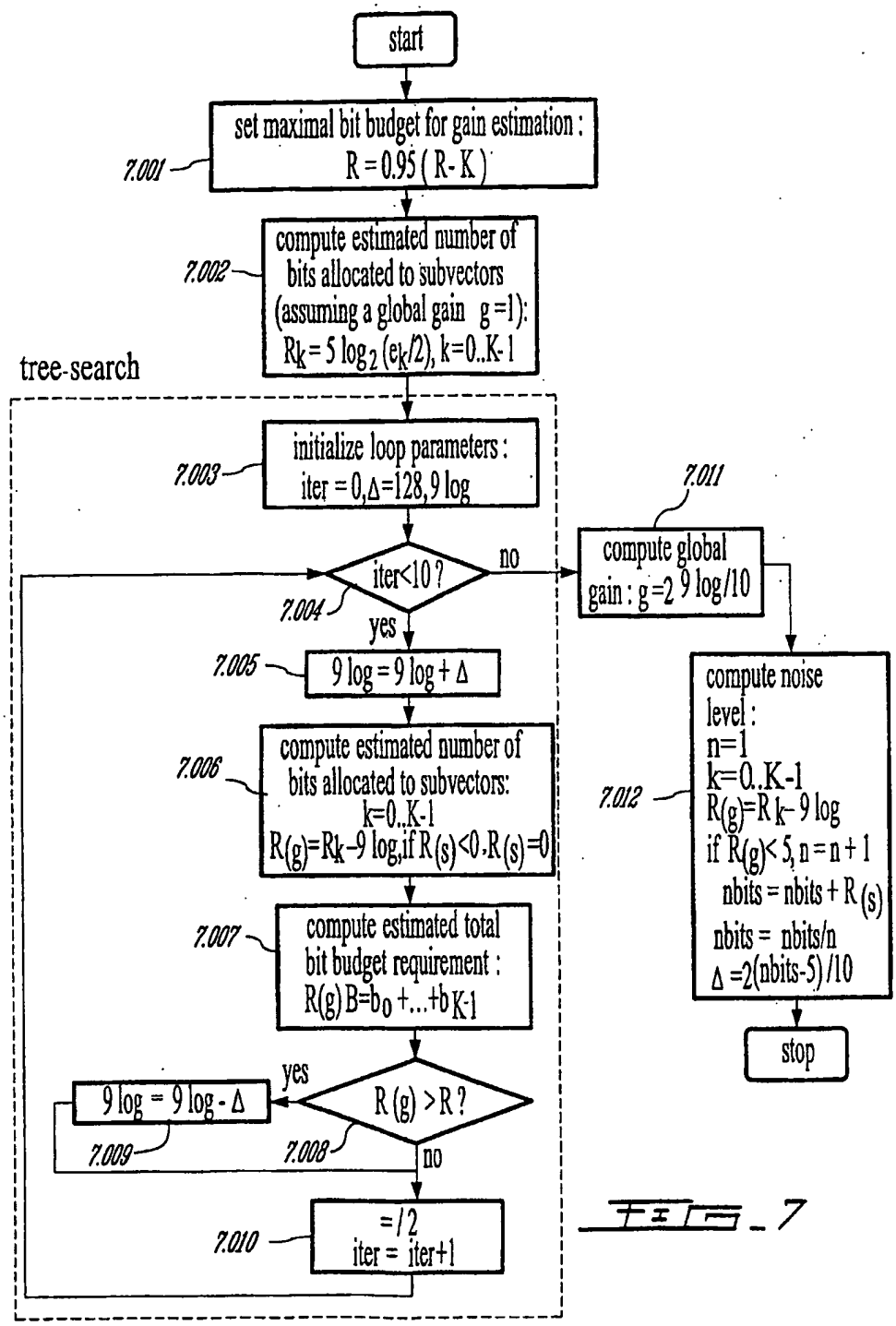

start
set maximal bit budget for gain estimation : R = 0.95 (R- K)
7.001
compute estimated number of bits allocated to subvectors (assuming a global gain g =1): Rk = 5 log2 (ek/2), k=0..K-1
7.002
tree-search
initialize loop parameters : iter = 0,Δ=128,9 log
7.003
iter<10 ?
7.004
no
yes
9 log = 9 log + Δ
7.005
compute estimated number of bits allocated to subvectors: k=0..K-1 R(g)=Rk-9 log,if R(s)<0,R(s)=0
7.006
compute estimated total bit budget requirement : R(g) B=b0 +...+bK-1
7.007
R (g) > R ?
7.008
yes
9 log = 9 log - Δ
7.009
no
=/2 iter = iter+1
7.010
7.011
compute global gain : g=2 9 log/10
7.012
compute noise level : n=1 k=0..K-1 R(g)=R k- 9 log if R(g)< 5, n = n + 1 nbits = nbits + R (s) nbits = nbits/n Δ =2(nbits-5) /10
stop

FIG. 7

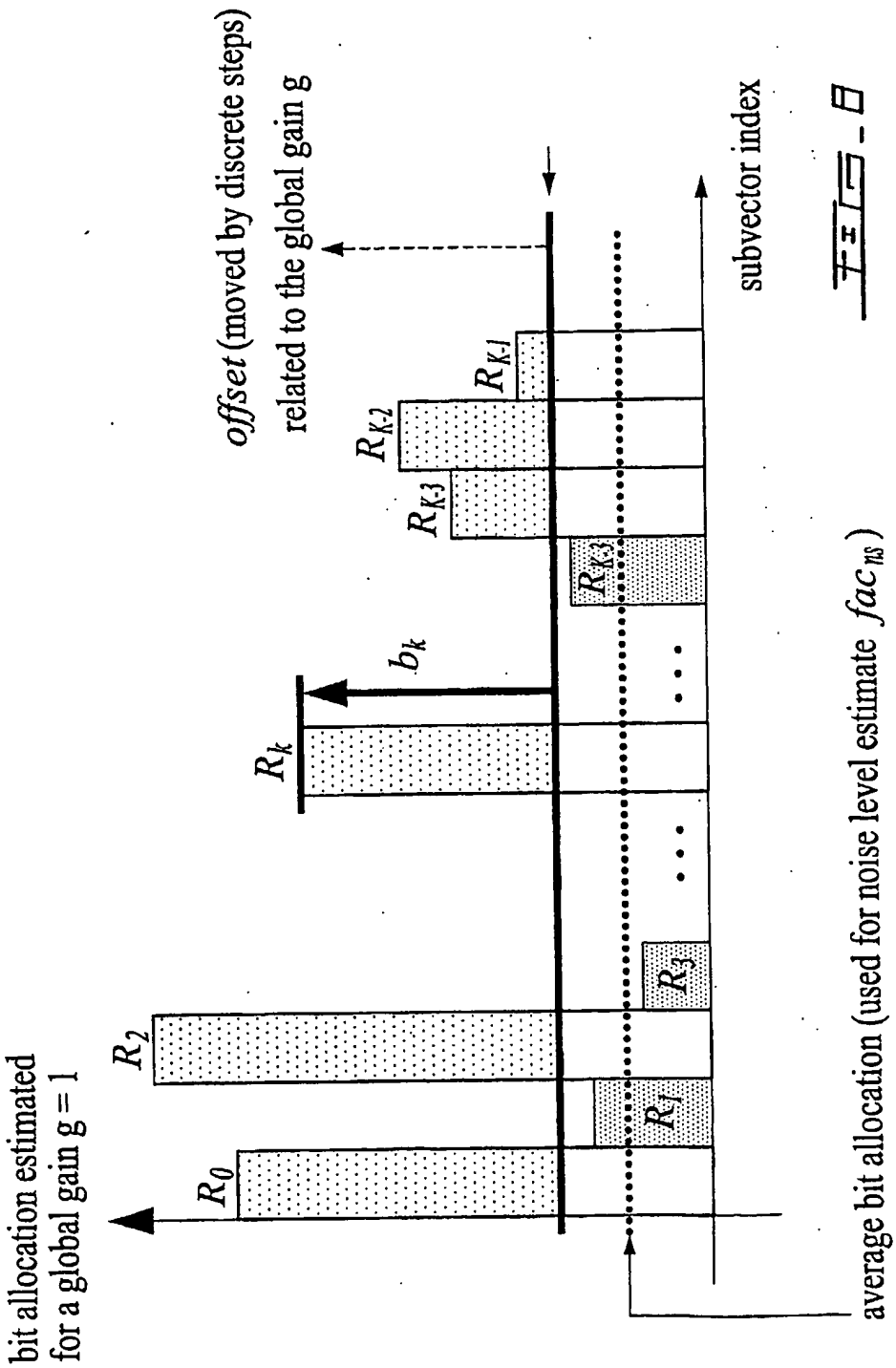

bit allocation estimated
for a global gain g = 1
R_0
R_2
R_k
b_k
offset (moved by discrete steps)
related to the global gain g
R_{K-1}
R_{K-2}
R_{K-3}
R_{K-3}
R_1
R_3
subvector index
average bit allocation (used for noise level estimate fac_{ns})

Fig - 8

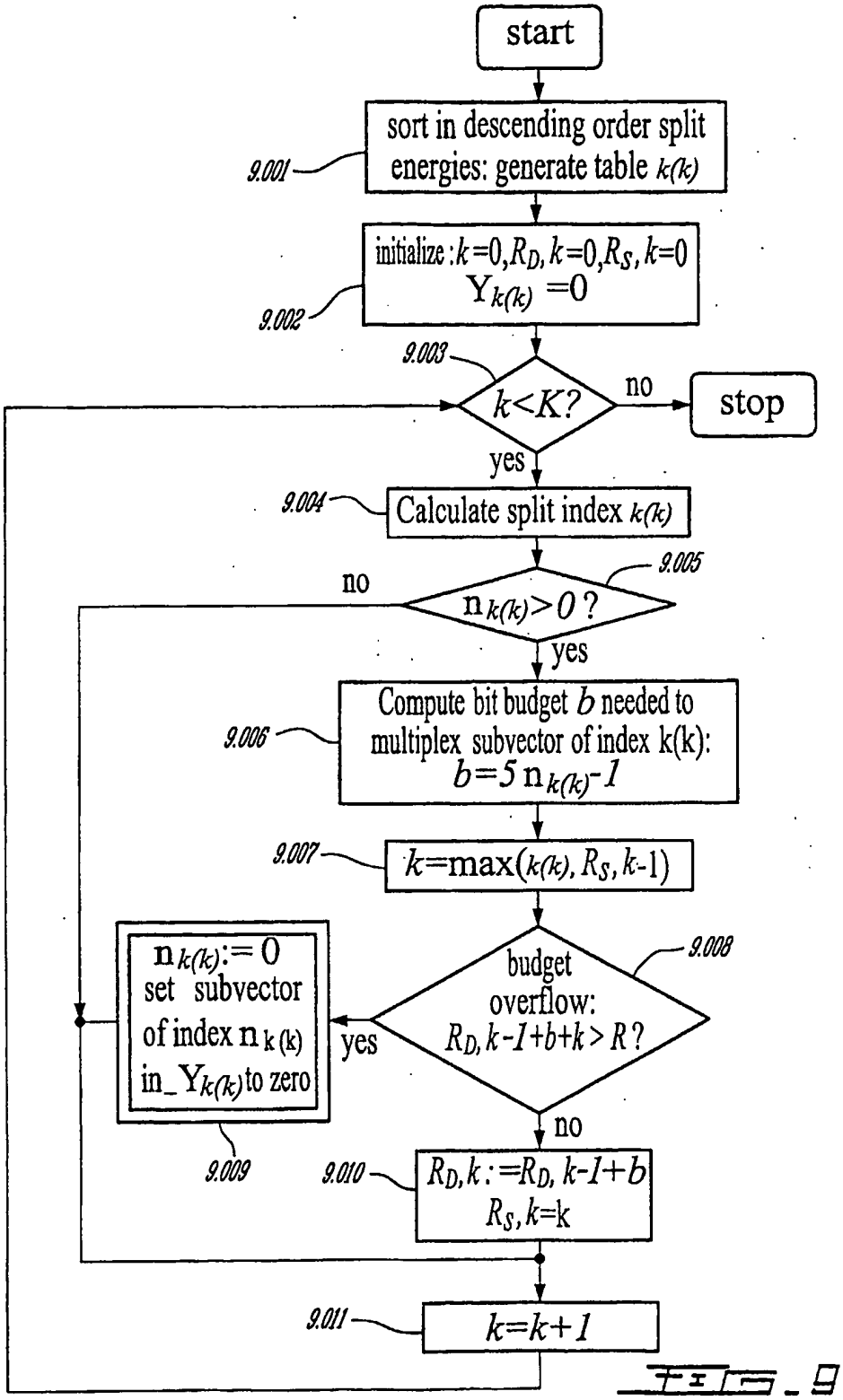

start
sort in descending order split energies: generate table k(k)
9.001
initialize :k=0,R_D,k=0,R_S,k=0
Y_k(k) =0
9.002
9.003
k<K?
no
stop
yes
9.004
Calculate split index k(k)
9.005
no
n_k(k)>0 ?
yes
Compute bit budget b needed to multiplex subvector of index k(k):
b=5 n_k(k)-1
9.006
9.007
k=max(k(k), R_S, k-1)
n_k(k):= 0
set subvector of index n_k(k) in_Y_k(k) to zero
9.009
budget overflow:
R_D,k-1+b+k > R ?
9.008
yes
no
9.010
R_D,k :=R_D, k-1+b
R_S,k=k
9.011
k=k+1

Fig. 9

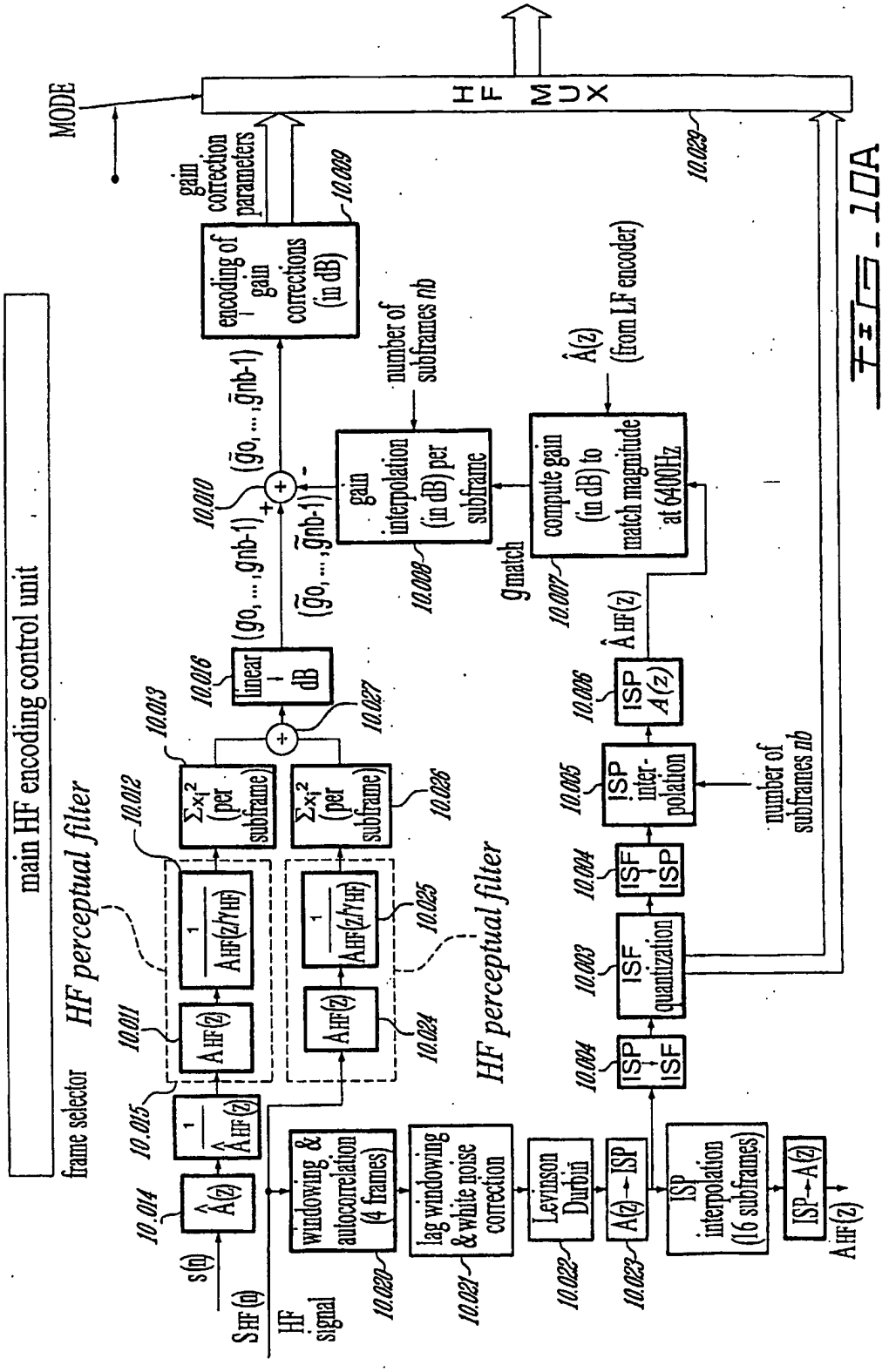

MODE
HF MUX
main HF encoding control unit
frame selector
HF perceptual filter
HF perceptual filter
gain correction parameters
encoding of gain corrections (in dB)
10.009
10.029
number of subframes nb
gain interpolation (in dB) per subframe
10.008
10.010
(g̃0, ..., g̃nb-1)
(g0, ..., gnb-1)
linear → dB
10.016
10.027
10.013
Σxi² (per subframe)
10.012
10.028
compute gain (in dB) to match magnitude at 6400Hz
10.007
gmatch
Â(z) (from LF encoder)
ÂHF(z)
ISP A(z)
10.006
ISP inter-polation
10.005
number of subframes nb
ISF ISP
10.004
ISF quantization
10.003
ISP ISF
10.004
Levinson Durbin
10.022
A(z) → ISP
10.023
ISP interpolation (16 subframes)
ISP→A(z)
AHF(z)
lag windowing & white noise correction
10.021
windowing & autocorrelation (4 frames)
10.020
HF signal
sHF(n)
s(n)
Â(z)
10.014
1/Â(z)
10.015
AHF(z)
10.011
1/AHF(z/γHF)
10.025
AHF(z)
10.024
1/AHF(z/γHF)
10.026

FIG - 10A

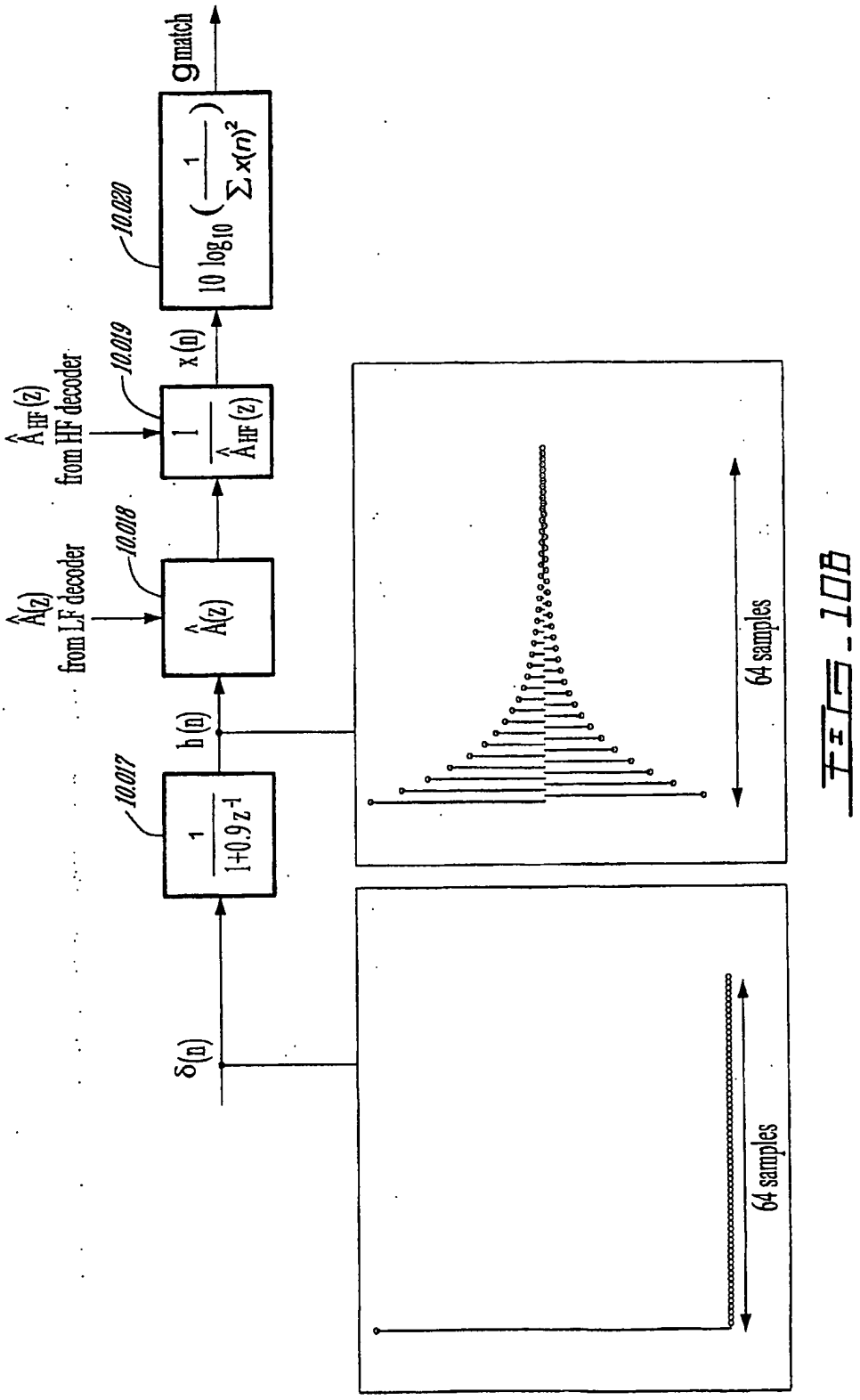

δ(n)
1/(1+0.9z⁻¹)
10.017
b(n)
Â(z)
from LF decoder
10.018
Â_HF(z)
from HF decoder
10.019
1/Â_HF(z)
x(n)
10 log10 (1/Σx(n)²)
10.020
gmatch
64 samples
64 samples

Fig. 10B

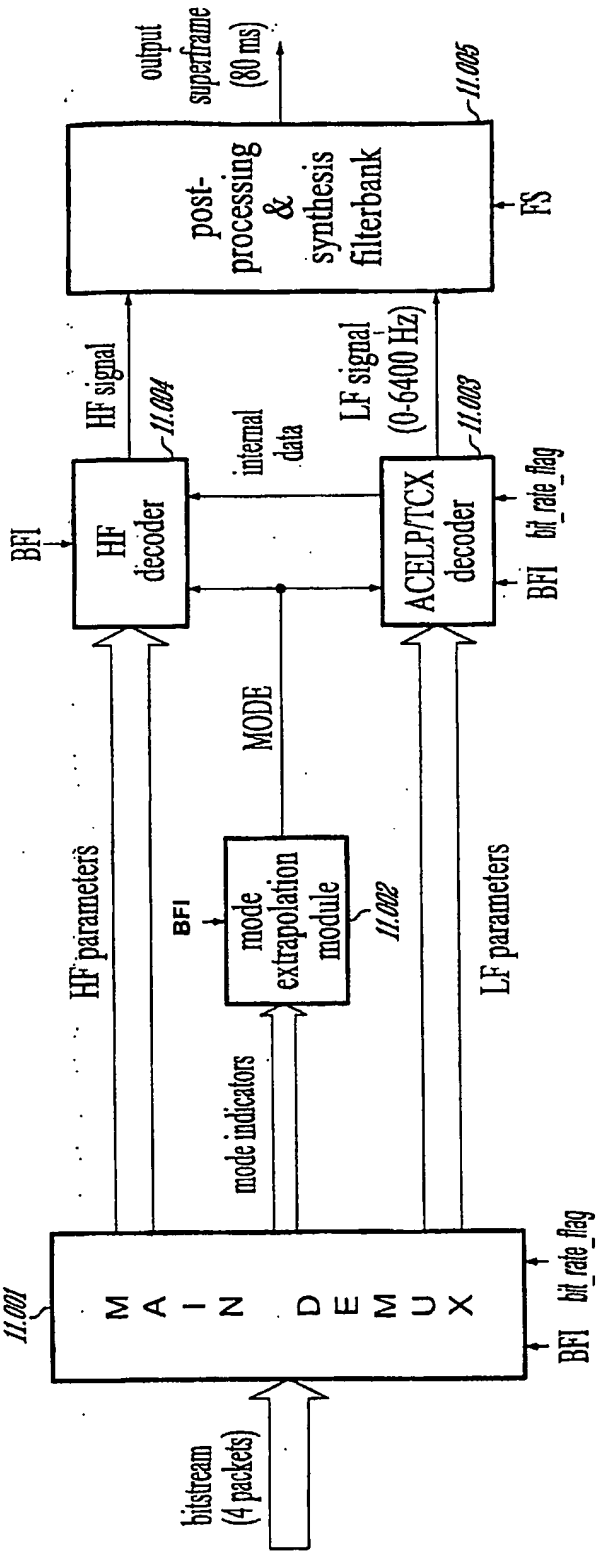

output superframe (80 ms)
11.005
post-processing & synthesis filterbank
FS
HF signal
11.004
LF signal (0-6400 Hz)
11.003
internal data
BFI
HF decoder
ACELP/TCX decoder
BFI bit_rate_flag
HF parameters
MODE
BFI
mode extrapolation module
11.002
LF parameters
mode indicators
11.001
MAIN DEMUX
BFI bit_rate_flag
bitstream (4 packets)

Fig. 11

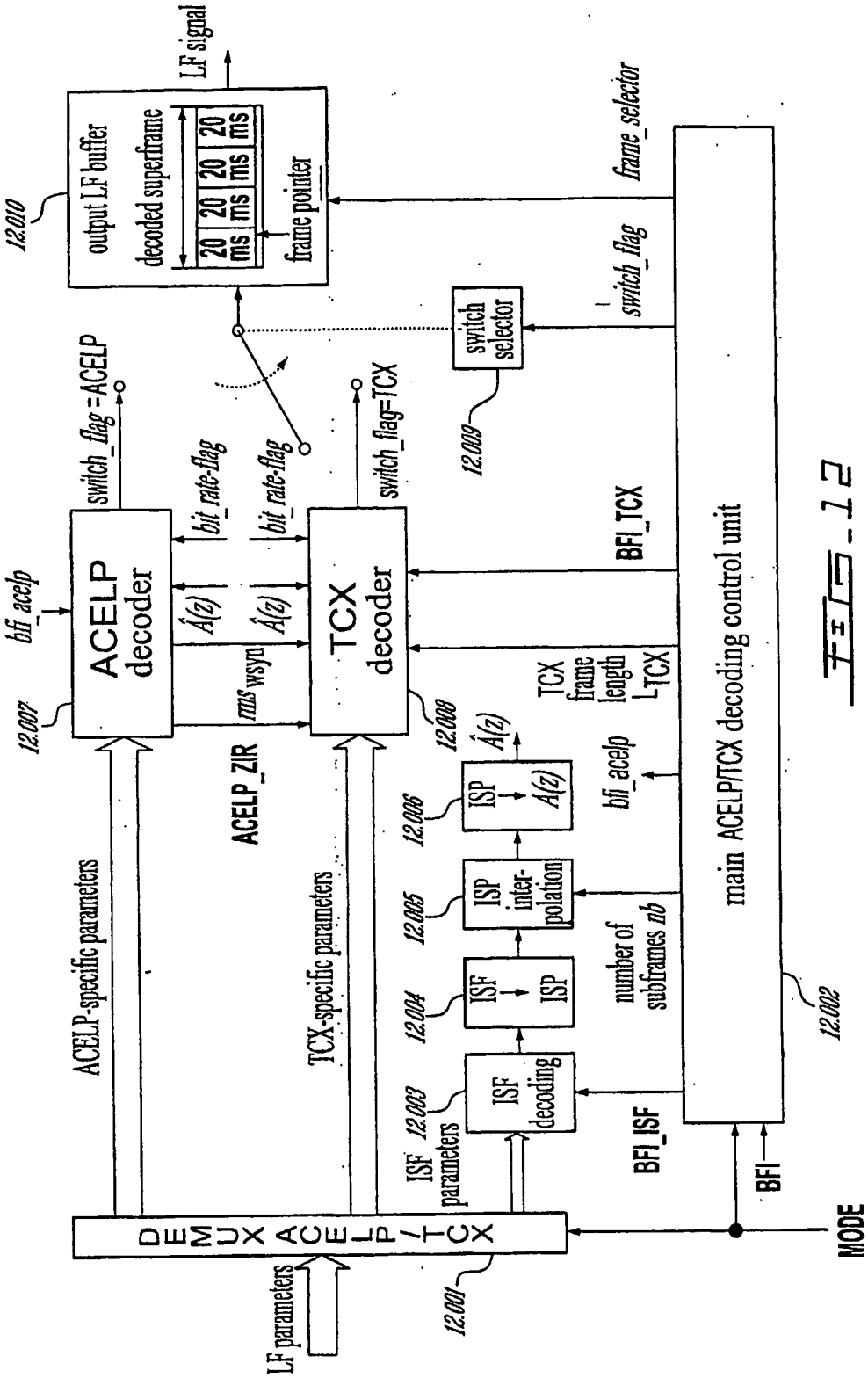

LF signal
output LF buffer
decoded superframe
20 ms
20 ms
20 ms
20 ms
frame pointer
12.010
frame_selector
switch_flag
switch selector
12.009
switch_flag=ACELP
switch_flag=TCX
bit_rate-flag
bit_rate-flag
bfi_acelp
ACELP decoder
TCX decoder
12.007
12.008
Â(z)
Â(z)
rms wsyn
ACELP_ZIR
BFI_TCX
TCX frame length LTCX
ISP → Â(z)
12.006
ISP interpolation
12.005
ISF → ISP
12.004
ISF decoding
12.003
ISF parameters
ACELP-specific parameters
TCX-specific parameters
number of subframes nb
bfi_acelp
main ACELP/TCX decoding control unit
12.002
BFI_ISF
BFI
MODE
DEMUX ACELP/TCX
LF parameters
12.001

FIG. 12

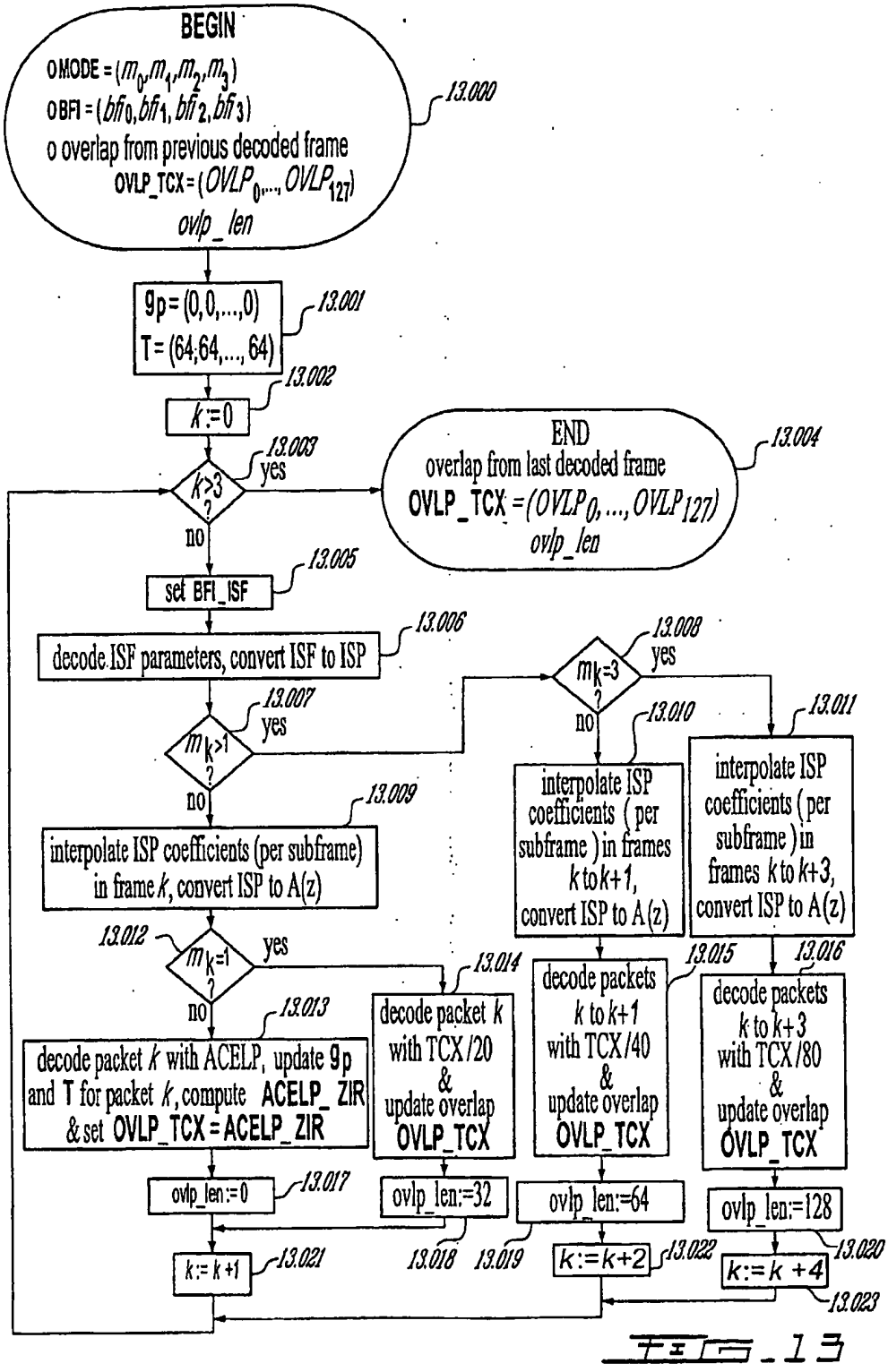

BEGIN
o MODE = (m_0, m_1, m_2, m_3)
o BFI = ($bfi_0, bfi_1, bfi_2, bfi_3$)
o overlap from previous decoded frame
OVLP_TCX = ($OVLP_0, ..., OVLP_{127}$)
ovlp_len
13.000
gp = (0,0,...,0)
T = (64;64,..., 64)
13.001
k := 0
13.002
k>3 ?
13.003
yes
no
END
overlap from last decoded frame
OVLP_TCX = ($OVLP_0, ..., OVLP_{127}$)
ovlp_len
13.004
set BFI_ISF
13.005
decode ISF parameters, convert ISF to ISP
13.006
$m_{k>1}$?
13.007
yes
no
$m_{k=3}$?
13.008
yes
no
interpolate ISP coefficients (per subframe) in frame k, convert ISP to A(z)
13.009
interpolate ISP coefficients (per subframe) in frames k to k+1, convert ISP to A(z)
13.010
interpolate ISP coefficients (per subframe) in frames k to k+3, convert ISP to A(z)
13.011
$m_{k=1}$?
13.012
yes
no
decode packet k with ACELP, update gp and T for packet k, compute ACELP_ZIR & set OVLP_TCX = ACELP_ZIR
13.013
decode packet k with TCX/20 & update overlap OVLP_TCX
13.014
decode packets k to k+1 with TCX/40 & update overlap OVLP_TCX
13.015
decode packets k to k+3 with TCX/80 & update overlap OVLP_TCX
13.016
ovlp_len:=0
13.017
ovlp_len:=32
13.018
ovlp_len:=64
13.019
ovlp_len:=128
13.020
k:= k+1
13.021
k:=k+2
13.022
k:=k+4
13.023

FIG. 13

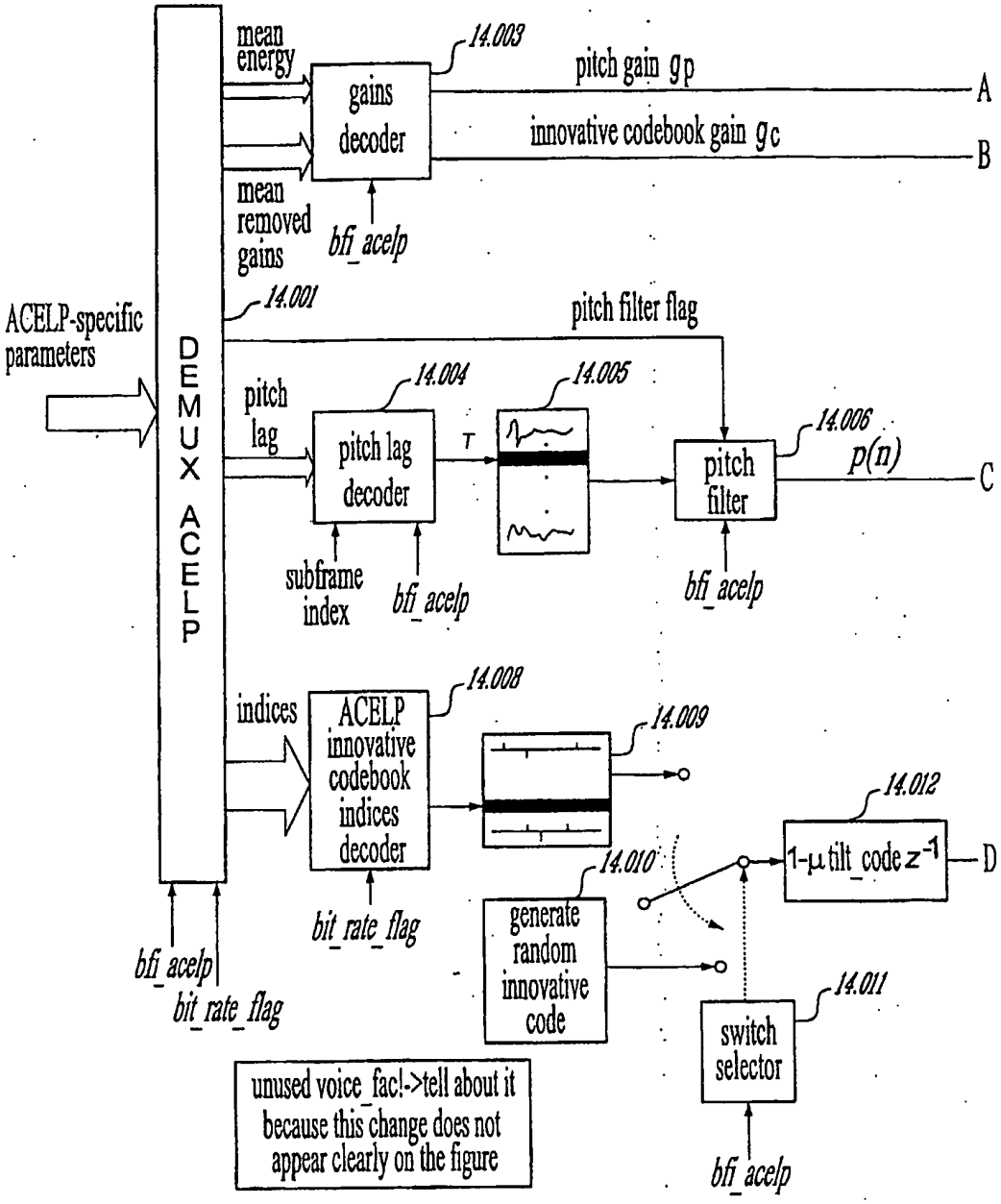

mean energy
14.003
gains decoder
pitch gain gp
A
innovative codebook gain gc
B
mean removed gains
bfi_acelp
ACELP-specific parameters
14.001
DEMUX ACELP
pitch filter flag
pitch lag
14.004
pitch lag decoder
T
14.005
14.006
pitch filter
p(n)
C
subframe index
bfi_acelp
bfi_acelp
indices
14.008
ACELP innovative codebook indices decoder
14.009
14.012
1-μ tilt_code z-1
D
14.010
generate random innovative code
bit_rate_flag
bfi_acelp
bit_rate_flag
14.011
switch selector
unused voice_fac!->tell about it because this change does not appear clearly on the figure
bfi_acelp

FIG_14A

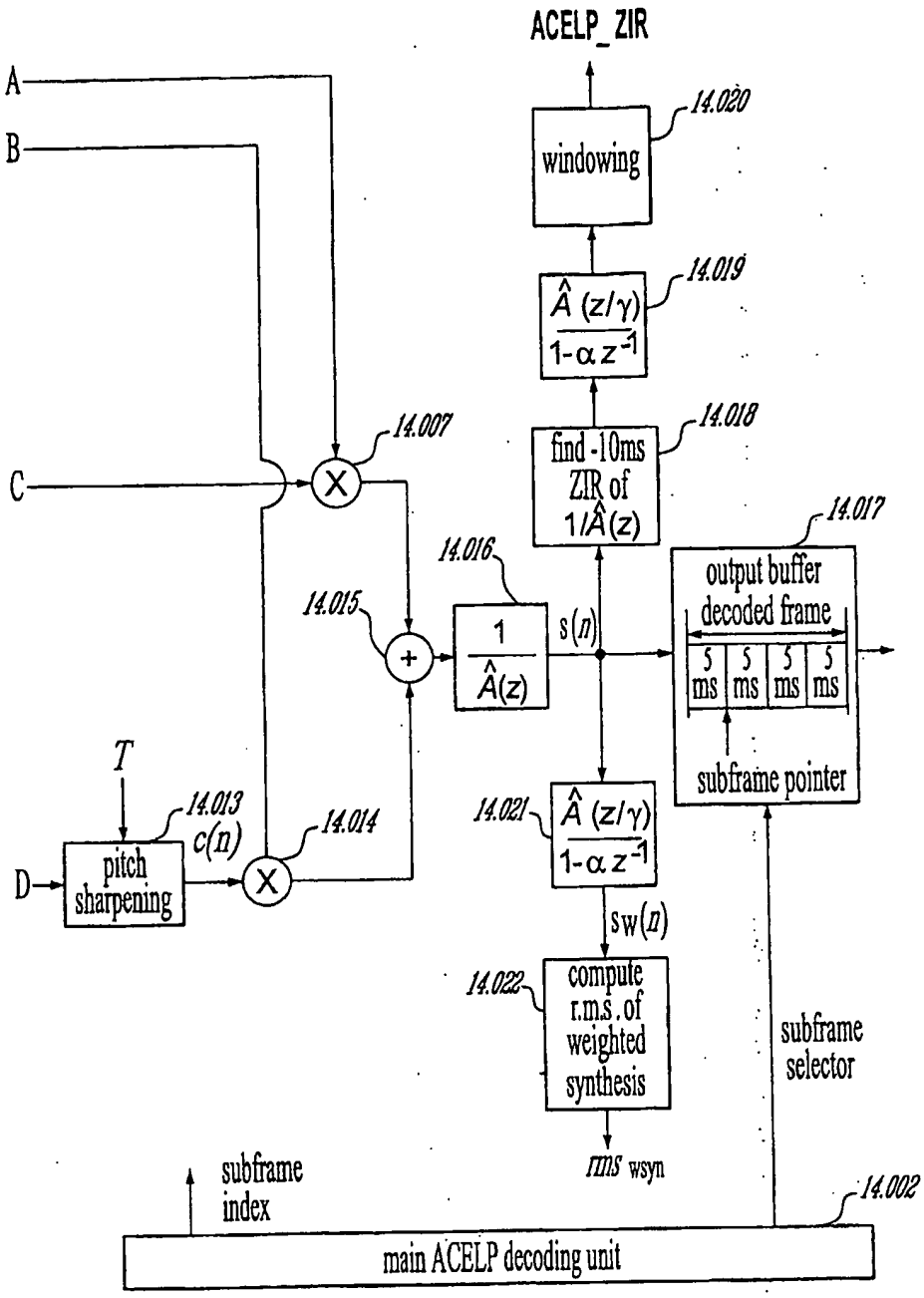

ACELP_ZIR
A
B
C
D
T
windowing
14.020
$\hat{A}(z/\gamma)$ / $1-\alpha z^{-1}$
14.019
find -10ms ZIR of $1/\hat{A}(z)$
14.018
14.007
14.016
14.015
$\frac{1}{\hat{A}(z)}$
s(n)
output buffer decoded frame
5 ms
5 ms
5 ms
5 ms
14.017
subframe pointer
14.013
c(n)
pitch sharpening
14.014
14.021
$\hat{A}(z/\gamma)$ / $1-\alpha z^{-1}$
sw(n)
14.022
compute r.m.s. of weighted synthesis
rms wsyn
subframe selector
subframe index
14.002
main ACELP decoding unit

FIG. 14B

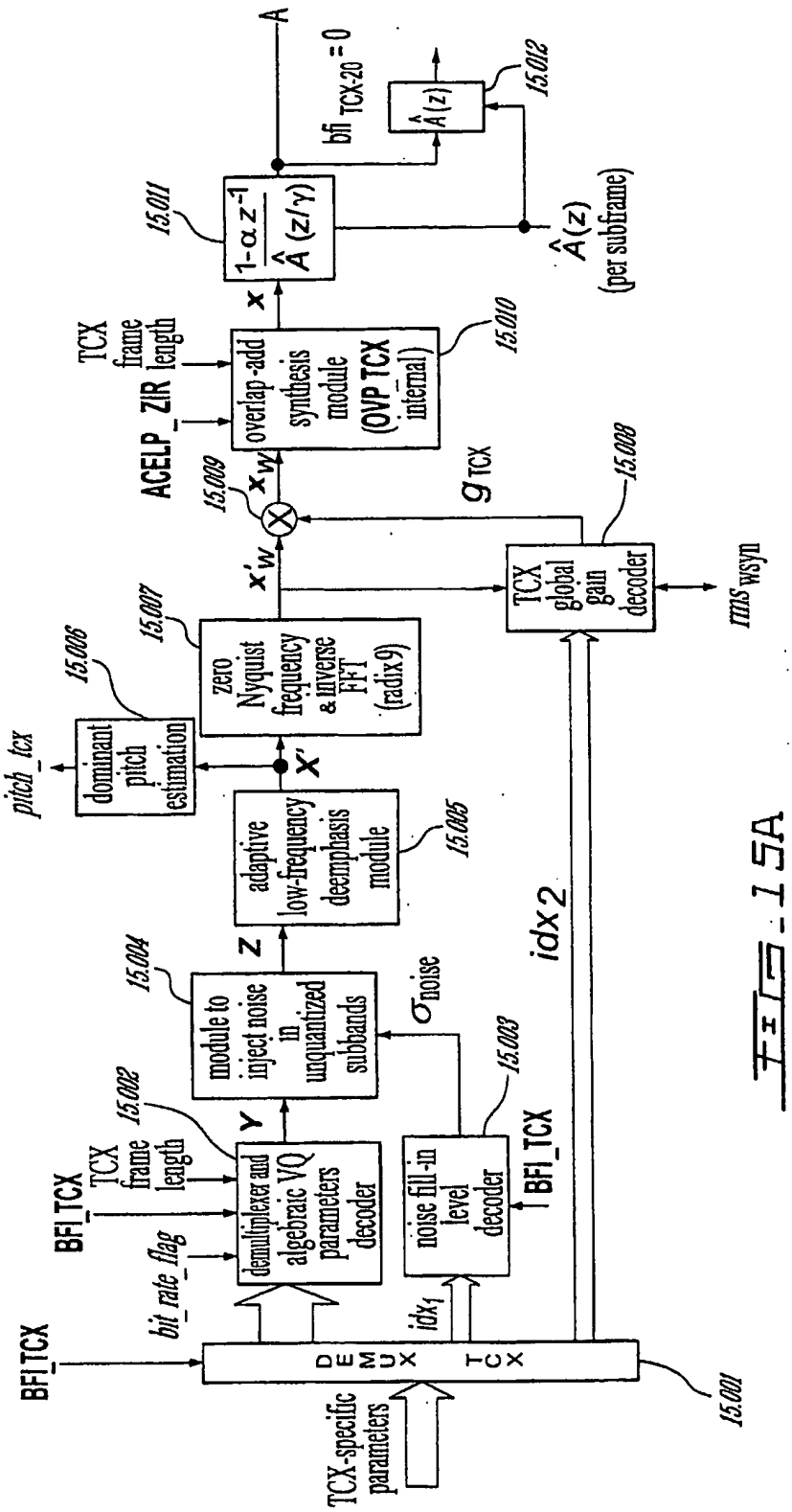

BFI_TCX
bit_rate_flag
BFI_TCX
TCX frame length
demultiplexer and algebraic VQ parameters decoder
15.002
noise fill-in level decoder
15.003
module to inject noise in unquantized subbands
15.004
adaptive low-frequency deemphasis module
15.005
dominant pitch estimation
15.006
zero Nyquist frequency & inverse FFT (radix 9)
15.007
TCX global gain decoder
15.008
overlap-add synthesis module (OVP TCX internal)
15.010
$\frac{1-\alpha z^{-1}}{\hat{A}(z/\gamma)}$
15.011
$\hat{A}(z)$
15.012
DEMUX
TCX
TCX-specific parameters
15.001
ACELP_ZIR
TCX frame length
pitch_tcx
idx1
idx2
σnoise
gTCX
rmswsyn
bfi TCX-20 = 0
A
Â(z)
(per subframe)
Y
Z
X'
x'w
xw
x

FIG. 15A

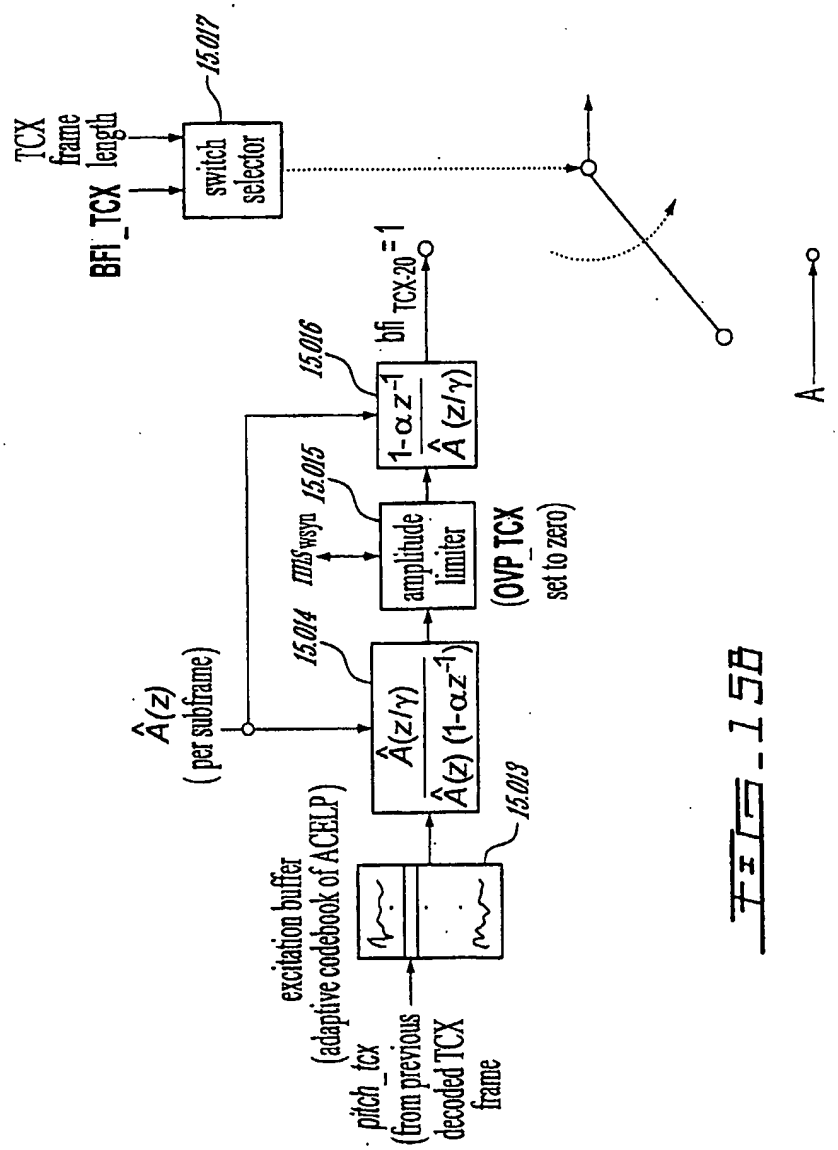

TCX frame length
BFI_TCX
switch selector
15.017
excitation buffer (adaptive codebook of ACELP)
pitch_tcx (from previous decoded TCX frame
15.013
$\hat{A}(z)$ (per subframe)
15.014
$\frac{\hat{A}(z/\gamma)}{\hat{A}(z)\ (1-\alpha z^{-1})}$
mds_{WSYN}
15.015
amplitude limiter
(OVP_TCX set to zero)
15.016
$\frac{1-\alpha z^{-1}}{\hat{A}(z/\gamma)}$
$bfi_{TCX-20}=1$
A

Fig. 15B

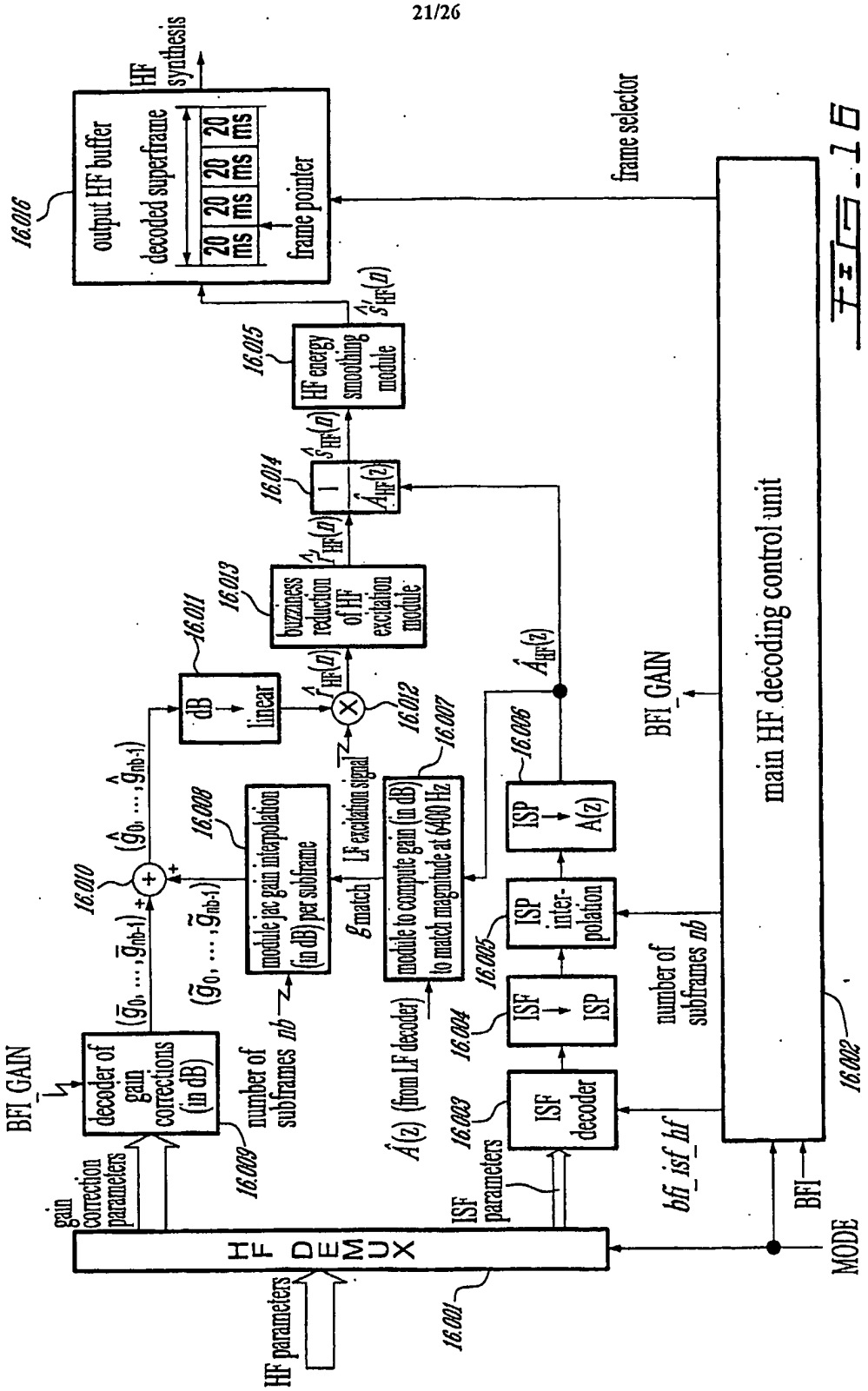

HF parameters
HF DEMUX
16.001
MODE
BFI
bfi_isf_hf
main HF decoding control unit
16.002
frame selector
BFI_GAIN
number of subframes nb
ISF parameters
ISF decoder
16.003
ISF → ISP
16.004
ISP interpolation
16.005
ISP → A(z)
16.006
Á(z) (from LF decoder)
module to compute gain (in dB) to match magnitude at 6400 Hz
16.007
g_match
module for gain interpolation (in dB) per subframe
16.008
number of subframes nb
gain correction parameters
decoder of gain corrections (in dB)
BFI_GAIN
16.009
16.010
dB → linear
16.011
LF excitation signal
16.012
buzziness reduction of HF excitation module
16.013
1/Á_HF(z)
16.014
HF energy smoothing module
16.015
output HF buffer
decoded superframe
20 ms
frame pointer
16.016
HF synthesis

FIG. 16

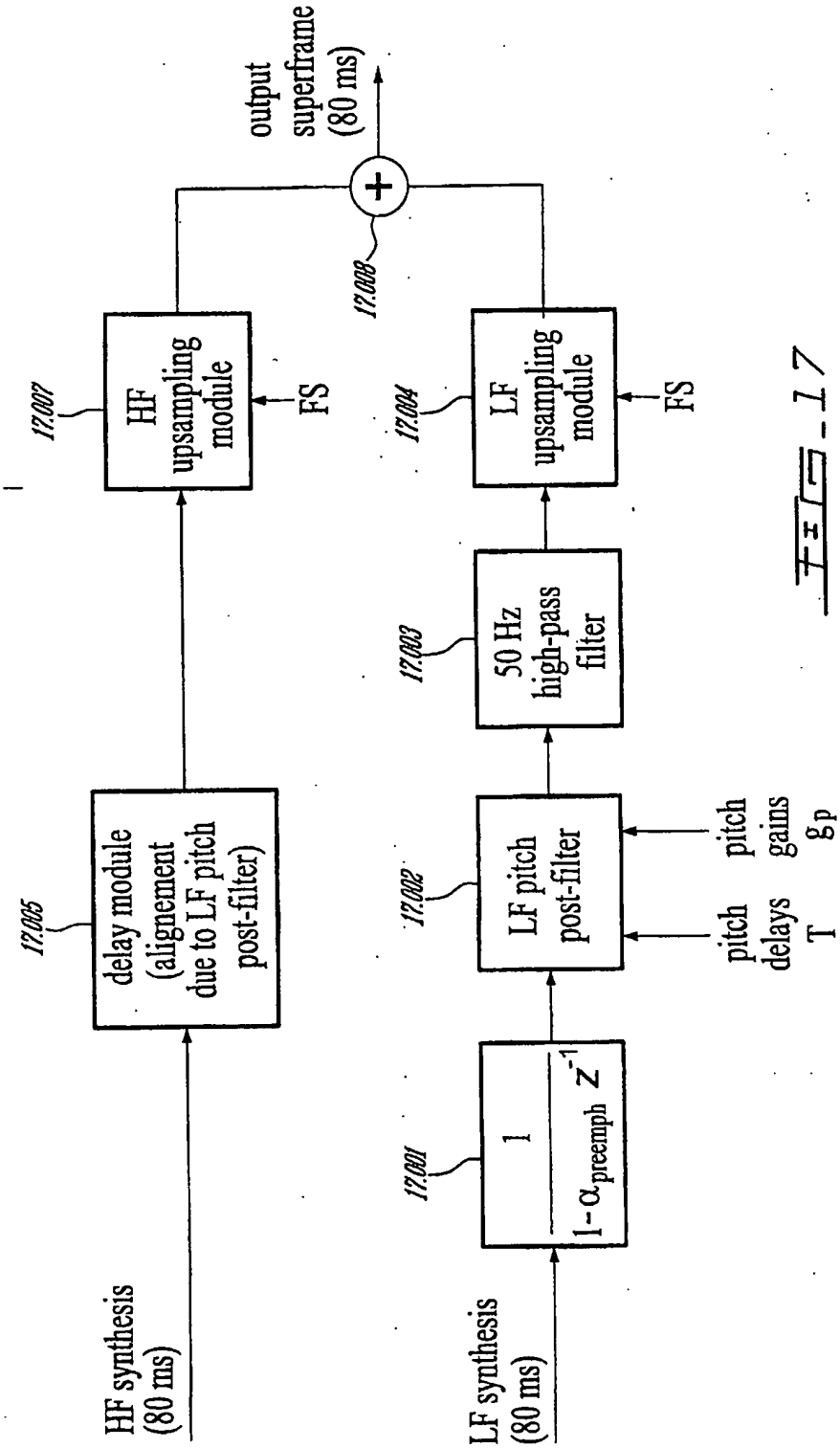

output superframe (80 ms)
17.008
17.007
HF upsampling module
FS
17.004
LF upsampling module
FS
17.005
delay module (alignement due to LF pitch post-filter)
17.003
50 Hz high-pass filter
17.002
LF pitch post-filter
pitch gains g_p
pitch delays T
17.001
$\frac{1}{1-\alpha_{preemph} z^{-1}}$
HF synthesis (80 ms)
LF synthesis (80 ms)

Fig. 17

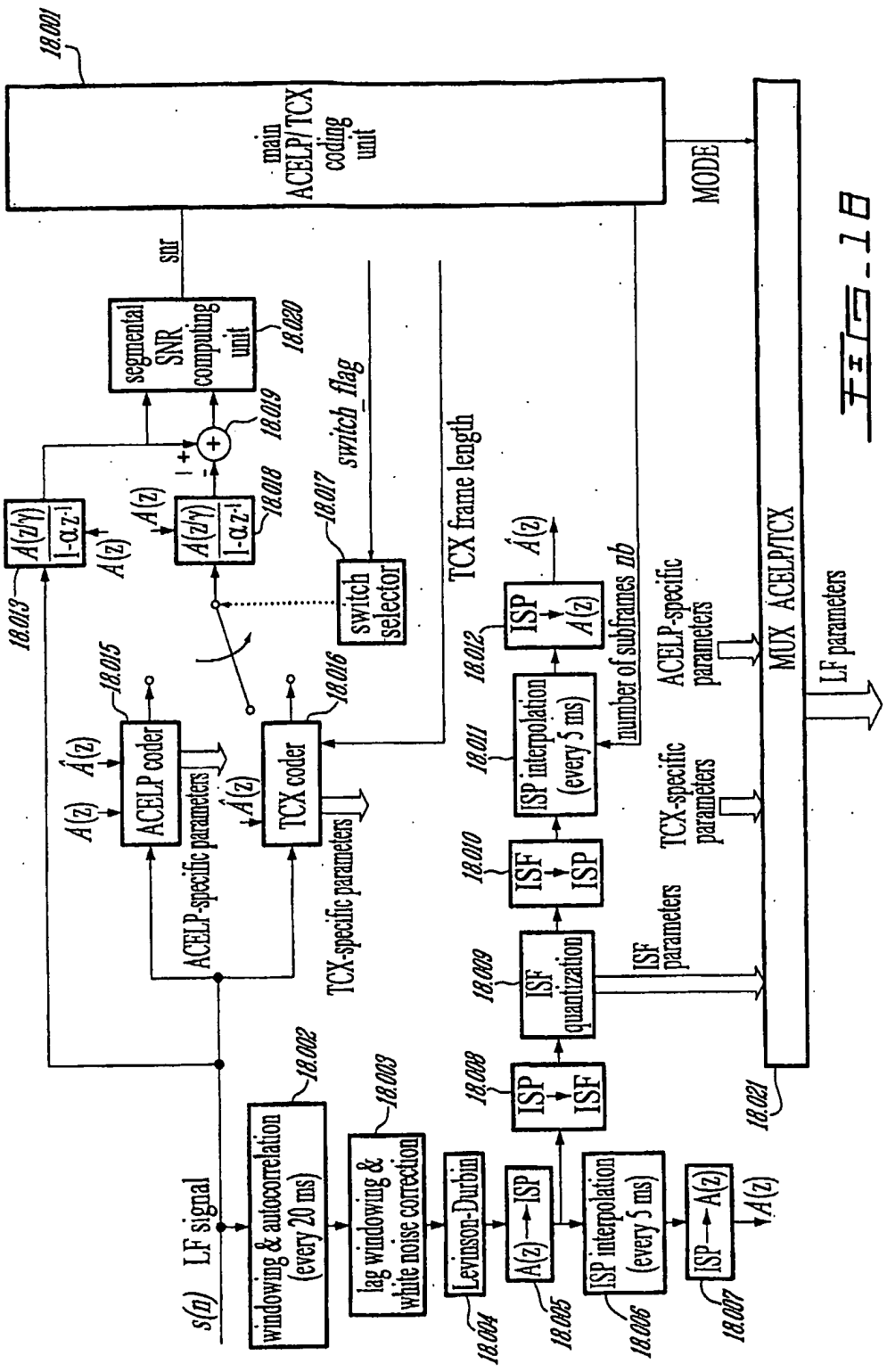

18.001
main ACELP/TCX coding unit
MODE
snr
segmental SNR computing unit
18.020
18.019
18.018
18.017
switch_flag
switch selector
TCX frame length
18.013
A(z/γ) / 1-αz⁻¹
A(z)
18.015
ACELP coder
18.016
TCX coder
ACELP-specific parameters
TCX-specific parameters
number of subframes nb
18.012
ISP → A(z)
Â(z)
18.011
ISP interpolation (every 5 ms)
18.010
ISF → ISP
18.009
ISF quantization
18.008
ISP → ISF
ISF parameters
ACELP-specific parameters
TCX-specific parameters
MUX ACELP/TCX
LF parameters
18.021
s(n) LF signal
18.002
windowing & autocorrelation (every 20 ms)
18.003
lag windowing & white noise correction
18.004
Levinson-Durbin
18.005
A(z) → ISP
18.006
ISP interpolation (every 5 ms)
18.007
ISP → A(z)
A(z)

FIG. 18

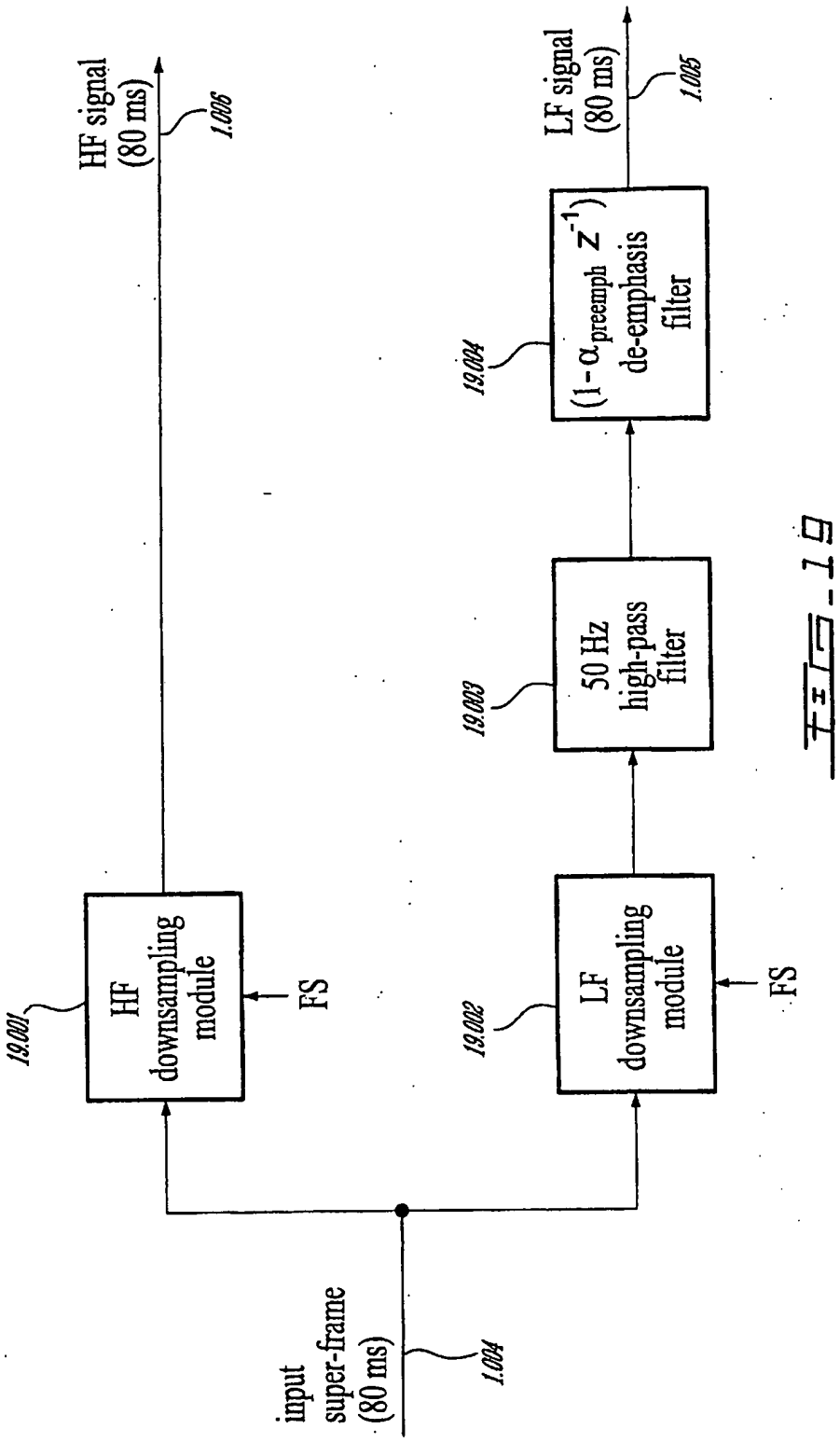

input super-frame (80 ms)
1.004
HF downsampling module
19.001
FS
HF signal (80 ms)
1.006
LF downsampling module
19.002
FS
50 Hz high-pass filter
19.003
(1- $\alpha_{preemph}$ z^{-1}) de-emphasis filter
19.004
LF signal (80 ms)
1.005

FIG - 19

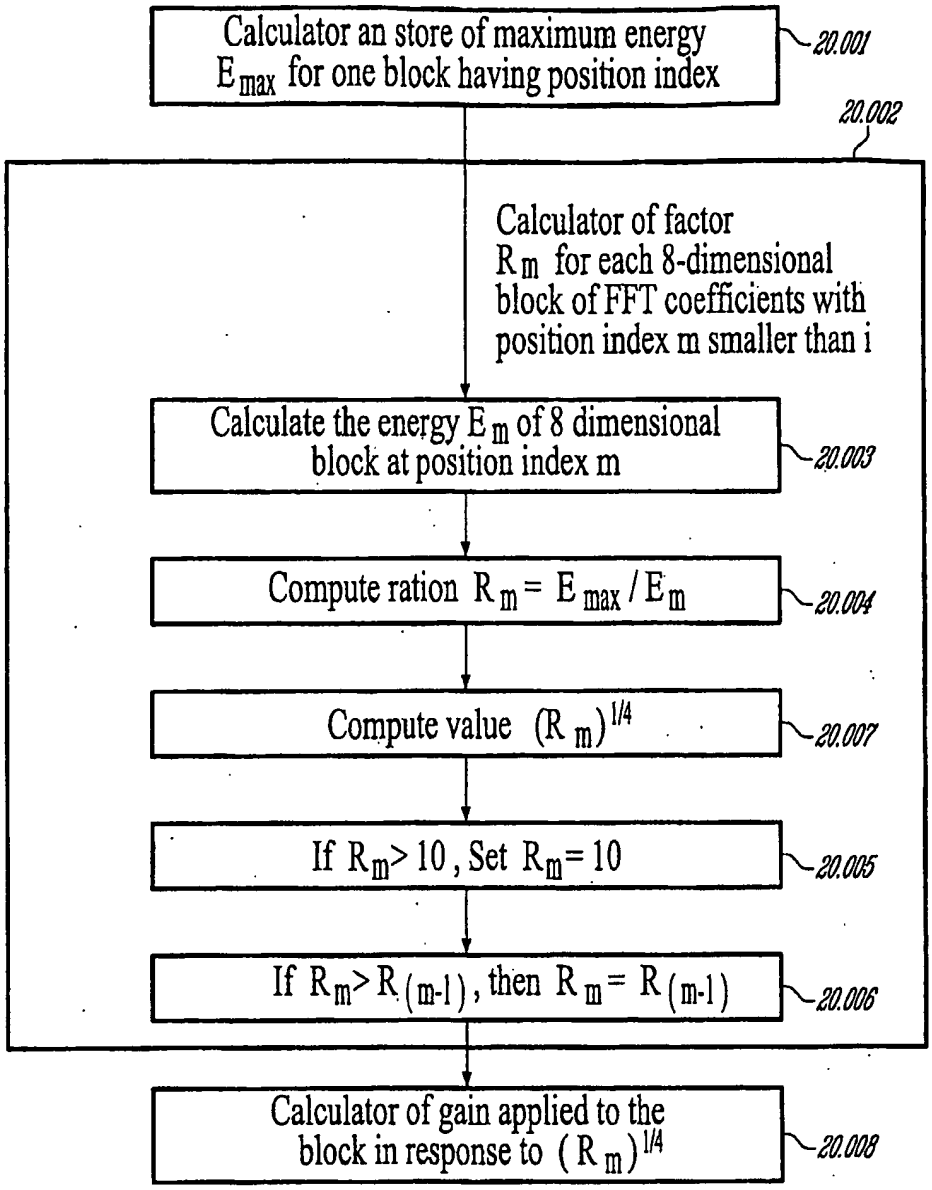

Calculator an store of maximum energy E_{max} for one block having position index
20.001
20.002
Calculator of factor R_m for each 8-dimensional block of FFT coefficients with position index m smaller than i
Calculate the energy E_m of 8 dimensional block at position index m
20.003
Compute ration $R_m = E_{max} / E_m$
20.004
Compute value $(R_m)^{1/4}$
20.007
If $R_m > 10$, Set $R_m = 10$
20.005
If $R_m > R_{(m-1)}$, then $R_m = R_{(m-1)}$
20.006
Calculator of gain applied to the block in response to $(R_m)^{1/4}$
20.008

Fig. 20

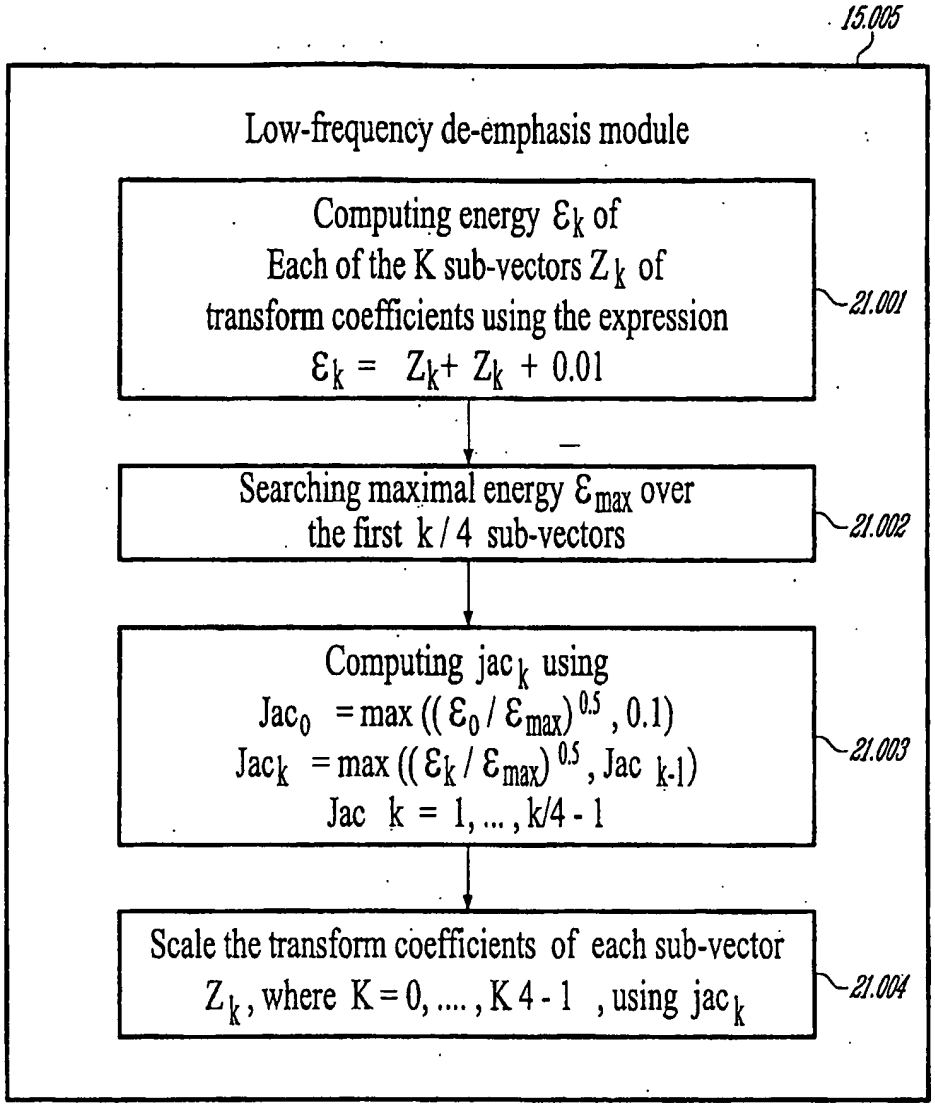

15.005
Low-frequency de-emphasis module
Computing energy ε_k of Each of the K sub-vectors Z_k of transform coefficients using the expression $\varepsilon_k = Z_k + Z_k + 0.01$
21.001
Searching maximal energy ε_{max} over the first k / 4 sub-vectors
21.002
Computing jac_k using $Jac_0 = \max((\varepsilon_0 / \varepsilon_{max})^{0.5}, 0.1)$ $Jac_k = \max((\varepsilon_k / \varepsilon_{max})^{0.5}, Jac_{k-1})$ Jac k = 1, ... , k/4 - 1
21.003
Scale the transform coefficients of each sub-vector Z_k, where K = 0, ... , K 4 - 1 , using jac_k
21.004

Fig. 21